

# Máster Avanzado de Programación en Python para Hacking, BigData y Machine Learning

CERTIFICACIÓN PCAP

# LECCIÓN 4

## Cadenas y Listas II

# ÍNDICE

- ✓ Introducción
- ✓ Objetivos
- ✓ Cadenas
- ✓ Listas Multidimensionales
- ✓ Comprensión de listas
- ✓ Operaciones con listas
- ✓ Conclusiones

# INTRODUCCIÓN

En esta lección vamos a continuar con el repaso del contenido del examen PCAP. Donde veremos el tipo cadena y continuaremos profundizando en el tipo lista que ya vimos en la lección anterior.

# OBJETIVOS

Al finalizar esta lección serás capaz de:

1

Conocer las características principales de las cadenas y los métodos y funciones para manipularlas

2

Conocer las características de las listas multidimensionales y cómo trabajar con ellas

3

Conocerla compresión de listas tanto unidimensionales como multidimensionales

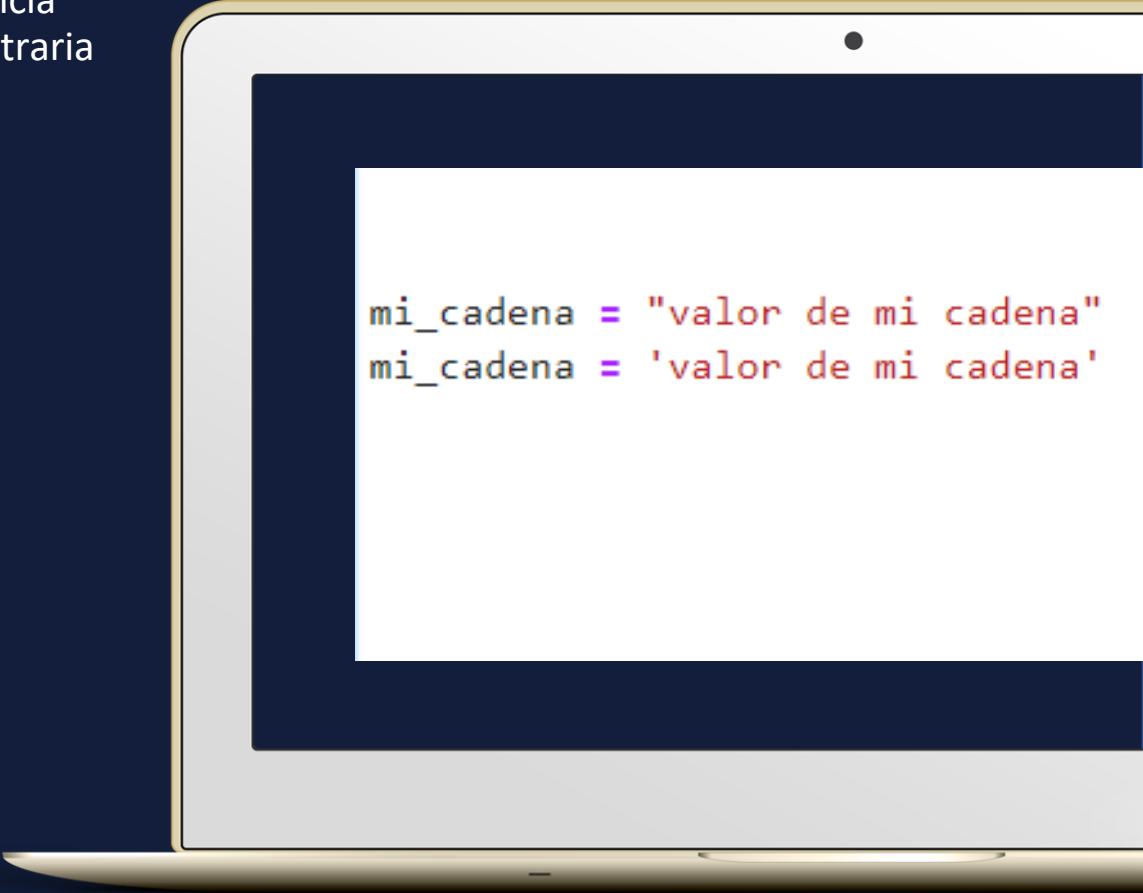
4

Conocer otras operaciones con listas como la búsqueda de elementos o la ordenación

## Cadenas: Definición

Una cadena en Python es una secuencia ordenada inmutable de longitud arbitraria pero finita de caracteres.

La longitud de una cadena se puede conocer con la función `len()`.



```
mi_cadena = "valor de mi cadena"  
mi_cadena = 'valor de mi cadena'
```

## Cadenas: Indexación



### Importante

- Se realiza igual que en las listas y tuplas.

```
mi_cadena = "valor de mi cadena"  
caracter = mi_cadena[indice]
```

## Cadenas: Seleccionando elementos de una lista (Slices)



### Importante

- Se realiza igual que en las listas y tuplas.

```
mi_cadena = "valor de mi cadena"  
mi_cadena[inicio:fin:salto]
```



## Cadenas: Inmutabilidad de las cadenas

Las cadenas son objetos inmutables, a diferencia de las listas.



- No se pueden cambiar los caracteres de la cadena: `TypeError`.
- Si asignamos a una variable de tipo cadena, una nueva cadena, cambiará la dirección de memoria a la que apunta.
- Si hacemos una concatenación entre cadenas, la cadena resultante será un nuevo objeto y su dirección de memoria también cambiará.

## Cadenas: Concatenación y replicación



### Importante

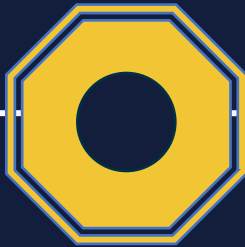
- Concatenación +
- Replicación: \*
- Ambos operadores se pueden usar en su versión abreviada.
- En la replicación num\_repeticiones tiene que ser entero sino TypeError.
- El resultado nuevo objeto de tipo cadena.

```
cadena1 = "valorcadena1"
cadena2 = "valorcadena1"
cadena3 = cadena1 + cadena2

cadena4 = cadena1 * num_repeticiones
```

## Cadenas: ord() y chr()

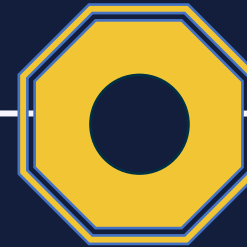
ord()



- Devuelve código de la tabla ASCII/Unicode.
- Entrada carácter sino error TypeError.

```
ord(caracter)
```

chr()



- Devuelve el carácter asociado al código indicado.
- Entrada entero sino error TypeError. Si entero mayor al rango de código de caracteres error ValueError.

```
chr(entero)
```

## Cadenas: Comparación de cadenas

Los operadores para comparar las cadenas son:

- ==
- !=
- <
- >
- <=
- >=

Operaciones para transformar tipos de datos:

- str() -> Convierte a cadena
- int() -> Convierte a entero
- float() -> Convierte a flotante

Si no se puede convertir error ValueError



### Importante

- Dos cadenas son iguales cuando están formadas por los mismos caracteres
- Los caracteres mayúscula tienen códigos inferiores a los caracteres minúscula.
- Para comparar dos cadenas diferentes python utiliza el primer caracteres diferente.
- Las cadenas que sólo contienen números se comparan carácter a carácter.
- Una cadena se puede comparar con un entero o float sólo con == y != y será siempre False.

## Listas Multidimensionales

Las listas multidimensionales son aquellas listas cuyos elementos son también listas y estos a su vez pueden contener también listas (matrices).



### Importante

- Las listas multidimensionales tienen las mismas propiedades y características de las listas con una dimensión.

```
lista_multi = [[valor11, valor12, ...],  
               [valor21, valor22, ...],  
               [valor31, valor32, ...]]
```

```
elemento = lista_multi[i][j]...
```

```
lista_multi[i][j] = nuevo_valor
```

## Listas Multidimensionales: Slices



### Importante

- Las listas multidimensionales permiten hacer slices al mismo tiempo de cada una de las dimensiones.

```
lista_multi = [[valor11, valor12, ...],  
               [valor21, valor22, ...],  
               [valor31, valor32, ...]]  
  
lista_multi[inicio:fin:paso][inicio:fin:paso]
```

## Compresión de listas: Listas Unidimensionales

Sintaxis básica



```
lista = [expresion for elemento in secuencia_iterable]
```

Sintaxis para añadir sólo algunos elementos



```
lista = [expresion for elemento in secuencia_iterable if condicion]
```

Sintaxis para sustituir algunos elementos



```
lista = [expresion if condicion else valor_def  
         for elemento in secuencia_iterable]
```

## Compresión de listas: Listas Multidimensionales

La compresión de listas multidimensionales puede parecer compleja pero es igual que en las listas unidimensionales.



```
lista = [[expresion for i in secuencia] for j in secuencia]

lista = [[expresion if condicion else valor_defecto for i in secuencia]
          for j in secuencia]
```



## Operaciones con listas: in y not in



### Importante

- in -> verifica si un valor está en la lista
- not in -> verifica si un valor no está en la lista
- in y not in se pueden también utilizar en cadenas, tuplas y diccionarios.
- En el caso de diccionarios se comprobará a nivel de clave.

```
elemento in lista  
elemento not in lista
```

## Operaciones con listas: ordenación



### Importante

- `sort()` ordena la lista en orden ascendente.
- `sort()` tiene parámetro `reverse` para cambiar el tipo de orden. Pasar por palabra clave.
- `sorted()` no modifica la lista que se le pasa.
- `reverse()` cambia el orden, no ordena.

```
mi_lista.sort()
mi_lista.sort(reverse=True)

mi_lista_ord = sorted(mi_lista)

mi_lista.reverse()
```



## CONCLUSIONES

1

Hemos visto las principales características, métodos y funciones de las cadenas

2

Hemos visto la definición de listas multidimensionales y cómo trabajar con ellas

3

Hemos visto como crear listas con las compresiones de listas, los operadores in y not in y la ordenación de listas

MUCHAS GRACIAS POR SU ATENCIÓN



[tcivera@grupomainjobs.com](mailto:tcivera@grupomainjobs.com)



Tamara Civera Lorenzo  
[es.linkedin.com/in/tamara-civera-lorenzo-95962147](https://es.linkedin.com/in/tamara-civera-lorenzo-95962147)



[twitter.com/eiposgrados](https://twitter.com/eiposgrados)



[facebook.com/eiposgrados](https://facebook.com/eiposgrados)



[instagram.com/eiposgrados](https://instagram.com/eiposgrados)